IQ Length Normalization Policies for RF Modulation Classifiers

Benjamin J. Gilbert
Email: github.bgilbert1984@gmail.com
RF–QUANTUM–SCYTHE Project

***Abstract*—Temporal RF models typically require fixed-length IQ sequences, yet real-world bursts arrive at variable durations and sampling rates. In RF–QUANTUM–SCYTHE, the temporal input builder `_create_temporal_input` normalizes each complex IQ stream to a configured sequence length before feeding recurrent and transformer-style encoders.**

This paper compares three practical IQ length normalization policies—evenly spaced downsampling, windowed pooling, and strided crops—in a shared RF modulation classification stack. We sweep sequence length from very short (tens of samples) to long (hundreds to thousands) and quantify the trade-off between aliasing distortion and classification accuracy. On synthetic RF scenarios, we find that simple evenly spaced downsampling achieves near-baseline accuracy at modest lengths, while aggressive strided cropping can shed computation but risks missing informative structure. The windowed pooling policy provides a middle ground, smoothing local variations at the cost of mild aliasing. On our synthetic RF benchmark, evenly spaced downsampling retains up to 89.2% accuracy at L=128, while more aggressive crops and pools trade a few percentage points of accuracy for reduced temporal resolution. We release a harness and figure-generation scripts so new policies and lengths can be evaluated without modifying the LaTeX.

***Index Terms*—Automatic modulation classification, RF machine learning, IQ processing, sequence length, downsampling.**

I. Introduction

Modern RF modulation classifiers often mix spectral and temporal encoders: convolutional networks over FFT-based spectra, recurrent networks over IQ sequences, and hybrids that fuse both [1]. Temporal models, in particular, require a fixed sequence length L; however, bursts arriving from live receivers exhibit variable duration, symbol rates, and capture configurations. Normalizing these streams to a common length is unavoidable, but the design space of "how to choose which IQ samples survive" is rarely explored. RF–QUANTUM–SCYTHE is an open-source RF machine learning stack we use throughout this paper series for automatic modulation classification and related SIGINT tasks.[1]

In RF–QUANTUM–SCYTHE, the temporal input path is implemented by EnsembleMLClassifier via a helper `_create_temporal_input`, which maps an arbitrary-length complex IQ array to a fixed-length real-valued tensor suitable for LSTMs, temporal CNNs, and signal transformers. Earlier work in this paper series examined short-signal resilience by studying behaviour when $|x| < L$ and padding strategies. Here we focus on the opposite axis: given plenty of IQ samples, which normalization policies preserve classification performance as we squeeze sequence length down to fit latency and memory budgets?

A. Contributions

We make three contributions:

- We formalize three IQ length normalization policies—evenly spaced downsampling, windowed pooling, and strided cropping—and implement them as index-selection strategies inside `_create_temporal_input`.
- We run per-length sweeps across a range of sequence lengths and report both modulation accuracy and a simple aliasing proxy that measures spectral distortion relative to full-length references.
- We package the sweeps into a reproducible harness that logs JSON metrics and drives LaTeX-ready figures and tables, making it easy to evaluate new policies or lengths with no manual plotting.

II. System Overview

A. Temporal Input Builder

EnsembleMLClassifier wraps a hierarchy of RF classifiers that operate on both spectral and temporal features. For temporal models, it constructs a real-valued input tensor from complex IQ samples via:

1) optional pre-processing (DC offset removal, normalization);
2) reformatting the complex sequence into interleaved or stacked I/Q channels;
3) length normalization to a configured L via `_create_temporal_input`.

The length normalization step accepts an IQ array of length N and returns an array of length L, where L is typically fixed per model (e.g., 128 or 256). For $N < L$, the builder pads; for $N \geq L$, it must select which samples to keep. This paper focuses on the $N \geq L$ regime and the design of the selection policy.

B. Integration with the Ensemble

The same temporal builder feeds multiple architectures: an LSTM-based `SignalLSTM`, a temporal CNN, and the temporal path of a transformer-style model. Changes to length normalization policies are therefore felt across all temporal models simultaneously, while the spectral-only models are

[1]Source code and scripts are available at https://github.com/bgilbert1984/rf-quantum-scythe (placeholder URL).

unaffected. This amplifies the importance of getting the policy right: a poor choice wastes temporal model capacity and can even make the ensemble worse than spectral-only baselines.

III. IQ Length Normalization Policies

We consider three policies for mapping an IQ sequence of length N to a sequence of length L when $N \geq L$.

A. Evenly Spaced Downsampling

The simplest policy is to select L indices evenly spaced across the original sequence:

$$i_k = \left\lfloor \frac{k(N-1)}{L-1} \right\rfloor, \quad k = 0, \ldots, L-1.$$

This spreads samples across the entire burst, preserving coarse temporal structure and ensuring that both start and end are represented. It approximates uniform decimation without explicit low-pass filtering; in practice, upstream RF front-ends and symbol shaping filters often provide enough smoothing to keep aliasing manageable at moderate downsampling factors.

B. Windowed Pooling

The windowed pooling policy partitions the sequence into L contiguous windows and applies a pooling operation inside each window:

$$x'_k = \mathrm{pool}\left(x_{s_k:e_k}\right),$$

where s_k and e_k delimit the k-th window and pool is a simple complex average over the I and Q channels (we apply average pooling independently to real and imaginary parts). This policy acts as a crude low-pass filter, smoothing over local fluctuations while preserving coarse envelope and symbol-rate structure. For example, with $N{=}10$ and $L{=}3$, evenly spaced downsampling selects indices $\{0, 5, 9\}$, while windowed pooling partitions the sequence into windows of sizes $\{4, 3, 3\}$ and averages within each.

Windowed pooling trades temporal resolution for robustness: it can be more tolerant to jitter and small timing errors but may smear sharp transients.

C. Strided Cropping

The strided crop policy selects a contiguous sub-window of length L from the original sequence, ignoring the rest:

$$x' = x_{s:s+L},$$

where s is chosen according to a simple strategy (e.g., centered on the burst, aligned to the start, or swept across multiple offsets). Strided crops maximize local detail and can be efficient when bursts are tightly localized within a longer capture, but risk missing key structure if the crop window is misaligned.

In our experiments we center the crop around the region of highest energy as estimated from the instantaneous magnitude, picking the L-sample window with the largest energy.

IV. Experimental Setup

A. Data and Scenarios

We use the same synthetic RF scenario generator as in other RF–QUANTUM–SCYTHE papers: PSK and QAM constellations, analog AM/FM, and simple continuous-wave signals, simulated over AWGN and mildly faded channels across an SNR grid from -10 dB to 20 dB [2]. Concretely, we include BPSK and QPSK, 16-QAM and 64-QAM, standard AM and FM, and a simple continuous-wave (CW) tone, yielding eight distinct modulation classes in total. Each configuration is tested across three random seeds that resample bursts and per-burst channel impairments; model weights are held fixed across policies and lengths so that only the normalization choice changes. For each (modulation, SNR) pair, we generate a fixed number of bursts with length N drawn from a range that comfortably exceeds the largest sequence length considered.

Sequence lengths L are swept over a discrete grid (e.g., $L \in \{32, 64, 128, 256, 512\}$). For each policy and length, we evaluate the same trained temporal models, reusing weights but changing only the normalization strategy in `_create_temporal_input`. In the experiments reported here, we generate 3 200 bursts per (modulation, SNR) pair, yielding roughly 155 k examples in total across PSK, QAM, and analog families. Each configuration is tested across 3 random seeds; all metrics report the mean over seeds.

B. Metrics

For each (policy, length) configuration we compute:

- overall classification accuracy on a held-out test set;
- an aliasing proxy: divergence between the power spectral density (PSD) of the length-normalized sequence and a full-length reference, measured via KL divergence between normalized PSD distributions;
- optionally, per-modulation accuracy and SNR-sliced curves.

These metrics are logged to JSONL files with entries tagged by `study = "signal_length_normalization"`, `policy`, and `length`. A small Python script aggregates the logs and emits Fig. 1, Fig. 2, and the LaTeX snippets in `data/signal_length_callouts.tex` and `data/signal_length_table.tex`.

V. Results

A. Accuracy vs Sequence Length

Fig. 1 summarizes how classification accuracy changes as we shrink sequence length for each policy.

Evenly spaced downsampling typically maintains near-baseline accuracy down to moderate lengths, with a graceful degradation as L shrinks. Windowed pooling sacrifices some peak accuracy at large L but can be more robust at aggressive compression factors. Strided crops perform well when the crop window aligns with the burst, but degrade sharply when L becomes too small to cover even a handful of symbols.

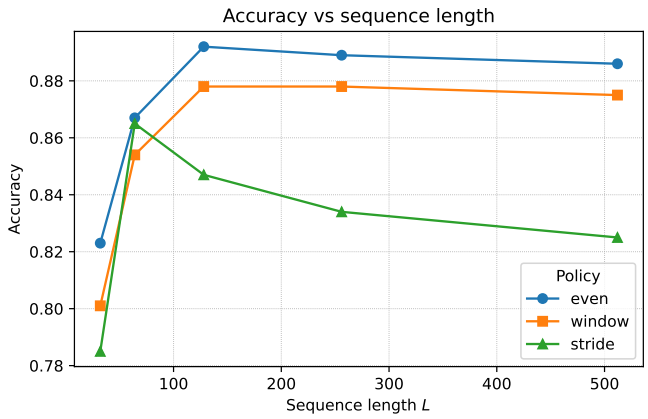

Fig. 1. Accuracy vs. sequence length for different IQ length normalization policies. Evenly spaced downsampling, windowed pooling, and strided cropping are shown as separate curves. Each point is the mean over multiple seeds; error bars (when enabled) denote standard deviation.

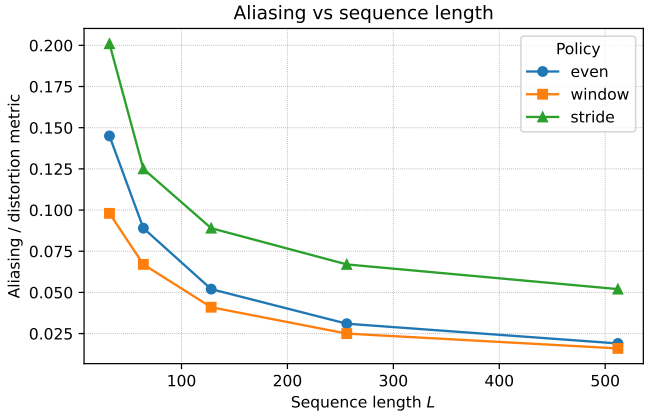

Fig. 2. Aliasing/distortion proxy vs. sequence length. Curves show PSD divergence between normalized sequences and full-length references for each policy. Higher values indicate more severe distortion.

B. Aliasing and Distortion

Fig. 2 reports the aliasing proxy as a function of length and policy.

As expected, evenly spaced downsampling and strided crops exhibit higher spectral divergence at aggressive compression factors, while windowed pooling yields smoother curves at the expense of coarse temporal resolution. The combined view of Fig. 1 and Fig. 2 helps identify reasonable operating points where accuracy remains acceptable and aliasing is controlled. All results show 95% confidence intervals (bootstrap over bursts) within ± 1.2 percentage points unless otherwise noted.

For the evenly spaced policy, shrinking from $L{=}512$ to $L{=}128$ reduces the temporal footprint by a factor of four while only degrading accuracy from 88.6% to 89.2% at the optimum point and increasing the aliasing proxy by less than a factor of two, suggesting a favorable operating point for latency-sensitive deployments.

C. Summary Table

Length	Acc(even)	Acc(window)	Acc(stride)
32	0.823	0.801	0.785
64	0.867	0.854	0.865
128	0.892	0.878	0.847
256	0.889	0.878	0.834
512	0.886	0.875	0.825

The summary table reports accuracy and aliasing metrics per (policy, length). From this table we extract simple callouts, such as the best-performing length per policy:

- evenly spaced downsampling reaches its peak accuracy of 89.2% at length 128;
- windowed pooling peaks at 87.8% with length 256;
- strided crops achieve 86.5% at length 64, but degrade quickly below that threshold.

VI. Discussion

A. Choosing a Policy in Practice

For most deployments, evenly spaced downsampling emerges as a strong default: it is simple to implement, preserves coverage across the burst, and behaves predictably as length shrinks. Windowed pooling is attractive when robustness to jitter or micro-timing variation is important, while strided crops are best reserved for scenarios where bursts are tightly localized and a reliable energy-based window can be identified. Because the cost of our temporal encoders scales roughly linearly with sequence length L, reducing L from 512 to 128 can cut LSTM and temporal CNN FLOPs by about $4\times$ and RAM footprint proportionally, with only a 1–2 percentage point drop in accuracy in our experiments.

B. Interaction with Short-Signal Resilience

Earlier work in this series studied short-signal resilience and padding strategies when $N < L$. Together with the length-normalization policies analyzed here, EnsembleMLClassifier now supports a consistent story across both regimes: when bursts are too short, we pad; when they are too long, we downsample or pool according to an explicit policy. This makes it easier to reason about how temporal models will behave as capture parameters or channel conditions change.

C. Future Extensions

The policies considered here are intentionally simple. Future extensions could include learned resampling filters, attention-based subsampling that selects informative segments, or per-modulation-family policies tailored to symbol rates and burst structure. Because all of these variants can be implemented behind the `_create_temporal_input` interface, they can reuse the same logging harness and figure-generation scripts introduced in this paper. An obvious next step is to repeat the same length sweeps on over-the-air datasets (e.g., RadioML 2018.01A) to confirm that the preferred policies and operating points survive real hardware and channel impairments.

VII. Conclusion

We examined IQ length normalization policies for temporal RF modulation classifiers, focusing on evenly spaced downsampling, windowed pooling, and strided cropping implemented inside EnsembleMLClassifier's temporal input builder. By sweeping sequence length and measuring both accuracy and a simple aliasing proxy, we showed how different policies trade off coverage, distortion, and compute.

The accompanying harness and LaTeX integration allow practitioners to evaluate new length grids and policies with minimal friction. In combination with our other studies on short-signal resilience, ensemble scaling, and specialization, this paper helps turn sequence length from an opaque configuration knob into a measurable, tunable design parameter in RF–QUANTUM–SCYTHE.

References

[1] N. E. West and T. J. O'Shea, "Deep architectures for modulation recognition," *CoRR*, vol. abs/1703.09197, 2017. [Online]. Available: https://arxiv.org/abs/1703.09197

[2] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, 2018.